

## 실제 구현 화면 및 코드

채팅 알고리즘 A001,A002

### 1. 채팅 준비

- 1) 웹 소켓 연결
- 2) 채팅 접속 후 메시지 구독시 설정
  - 읽음 여부를 1로 업데이트하여 읽음 처리, 특정 상대와의 대화 이력들을 가져옴
- 3) 세션(고객/업체)에 따라 고객 모드 ON/OFF 설정
- 4) 고객/업체 정보를 가져와서 데이터 바인딩
- 5) 최근순으로 대화 목록을 가져옴
- 6) 읽지 않은 메시지 개수를 가져옴
- 7) 화면에 표시

### 2. 채팅 화면 처리

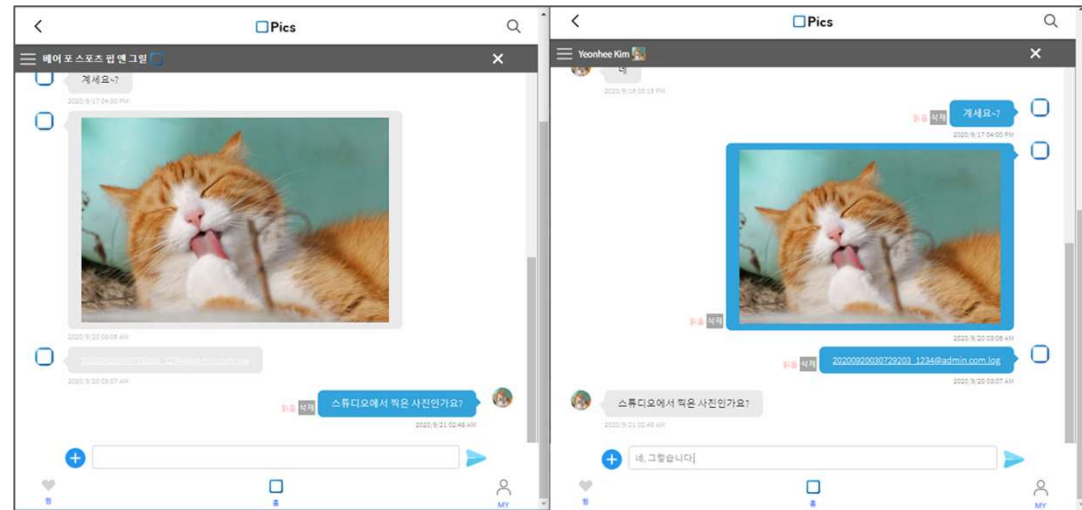
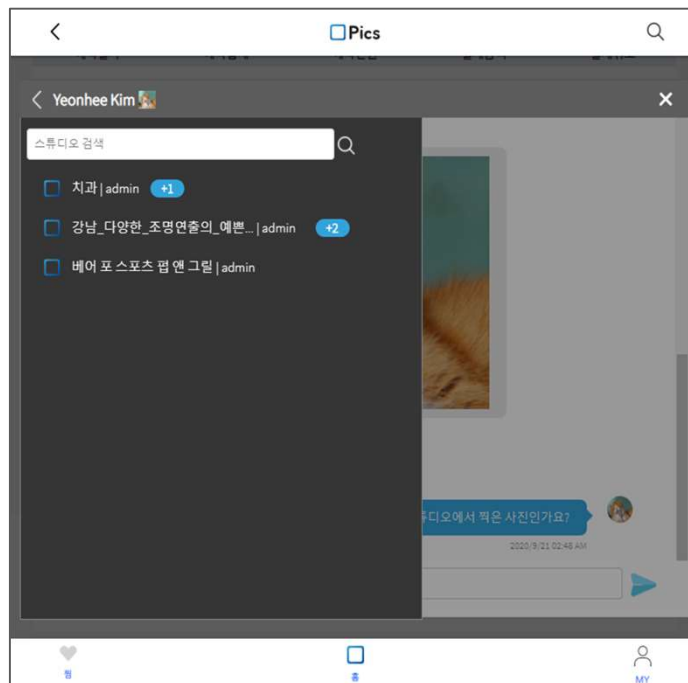
- 1) 스튜디오 이름, 고객 닉네임, 문의 내용 글자수는 15자까지만 표시
- 2) 문의 날짜는 연/월/일/시/분 오전/오후로 표시
- 3) 같은 시간일 경우 앞선 대화의 시간을 보이지 않음
- 4) 대화를 불러오거나 대화가 추가되면 스크롤을 최하단으로 내림

## 실제 구현 화면 및 코드

<p>3. 실시간 1대1 채팅</p>	<p>1) 채팅 Modal 활성화시 채팅 구독 여부 ON 표시</p> <p>2) 채팅 기본 설정</p> <ul style="list-style-type: none"> <li>- 대화 선택 여부 ON</li> <li>- 고객 아이디와 스튜디오 아이디 데이터 바인딩</li> <li>- 고객/업체 여부에 따라 발신 표시를 0 또는 1로 데이터 바인딩</li> <li>- 1-2)에서와 같이 읽음 처리 후 대화 상대와의 이때까지 대화 이력을 가져옴</li> </ul>
<p>Pub/Sub</p> <p>4. 채팅 발신 (메세지 발행)</p> <p>5. 채팅 수신 (메세지 구독)</p>	<p>1) 채팅 전송</p> <ul style="list-style-type: none"> <li>- 텍스트 : 바로 전송</li> <li>- 파일 : 5MB 이내 제한, 파일이름 및 경로 화면 출력, 확인 후 전송</li> <li>- 전송된 채팅은 고객/업체 발신 표시, 텍스트/파일 여부에 따라 DB에 등록</li> </ul> <p>2) 전송 후 화면 처리</p> <ul style="list-style-type: none"> <li>- 텍스트 및 파일 : 입력 리셋</li> <li>- 파일 : 파일 업로드, 파일 첨부 Modal 숨김</li> </ul> <p>1) 채팅 수신</p> <ul style="list-style-type: none"> <li>- 해당되는 고객, 업체에게만 1대 1 채팅 응답 결과가 수신됨</li> <li>- 채팅 구독 여부가 ON인 경우에만 수신됨(채팅창을 나간 경우에는 수신되지 않음)</li> <li>- 1-2)에서 설정한대로 읽음 처리 후 대화 상대와의 이때까지 대화 이력을 가져옴</li> </ul> <p>2) 수신 후 화면 처리</p> <p>파일 : 파일 확장자 체크 → 이미지 파일은 화면에 출력 / 일반 파일은 링크로 출력</p>
<p>6. 채팅 종료</p>	<p>채팅 Modal 숨김시 채팅 구독 여부 OFF 표시</p>

## 실제 구현 화면 및 코드

### 채팅 구현 화면



#### 문의내역

스튜디오(업체명)	내용	날짜
치과(admin)	비밀입니다. +1	2020/9/20
강남_다양한_조명연출의_예쁜... (admin)	기다리겠습니다. +2	2020/9/20
베어 포 스포츠 펄 앤 그릴(admin)	스튜디오에서 찍은 사진인가요...	2020/9/21

## 실제 구현 화면 및 코드

채팅 코드 A001,A002

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    /* 클라이언트가 메시지를 구독할 endpoint를 정의 */
    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker("/send");
        config.setApplicationDestinationPrefixes("/"); //도착경로에 대한 prefix 설정 부분
    }

    /* connection을 맺을 때 CORS 허용 */
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/websocket").setAllowedOrigins("*").withSockJS();
    }
}
```

웹소켓 설정 (Back)

```
@MessageMapping("/receive") //receive를 메시지를 받을 endpoint로 설정
@SendTo("/send") //send로 메시지를 반환
public Chat ChatHandler(@RequestBody Chat chat) {
    Chat resultChat = new Chat();
    try {
        int result = chatService.addChat(chat);
        Map map = new HashMap();
        map.put("custId", chat.getCustId());
        map.put("stuId", chat.getStuId());
        resultChat = chatService.getMostRecentChat(map);
    } catch (Exception e) {
        //e.printStackTrace();
    }
    return resultChat;
}
```

채팅 발행 및 구독 (Back)

## 실제 구현 화면 및 코드

채팅 코드 A001,A002

```
/* 웹소켓 연결 */
connect() {
  const serverURL = "http://localhost:7777/webSocket"
  let socket = new SockJS(serverURL);
  this.stompClient = Stomp.over(socket);
  console.log(`소켓 연결 시도... 서버 주소: ${serverURL}`)
  this.stompClient.connect({},
    frame => { /* 연결 성공 */
      this.connected = true;
      /* 서버의 메시지 전송 endpoing를 구독(Pub/Sub 구조) */
      this.stompClient.subscribe("/send", response => {
        if (this.chat.stuId == JSON.parse(response.body).stuId &&
            this.chat.custId == JSON.parse(response.body).custId &&
            this.chatSubscribe) {
          console.log('구독으로 받은 메시지 : ', response.body);
          this.updateReadCheck(); //읽음 처리 -> 대화 가져오기
        }
      });
    },
    error => { /* 연결 실패 */
      this.connected = false;
    }
  );
},
```

웹소켓 연결 및 구독 설정(Front)

```
/* 메시지 전송 */
send() {
  console.log("보내는 메시지:" + this.chat.word);
  console.log("보내는 파일:" + this.chat.filePath);
  if (this.stompClient && this.stompClient.connected) {
    const msg = this.chat;
    this.stompClient.send("/receive", JSON.stringify(msg), {});
    this.controlModal('hide', 'uploadModal');

    /* 보내고 나서 입력 리셋 */
    this.chat.word = '';
    this.chat.filePath = '';
    document.getElementById('chatFile').value = '';
    document.getElementById('chatFileName').value = '';
  }
},
```

메세지 발행 및 입력 리셋(Front)

## 실제 구현 화면 및 코드

### 예약 알고리즘 (S002)

입력값 제한	날짜 선택 제한	- 오늘 날짜 < 예약 시작일 < 예약 종료일
	시간 선택 제한	- 1일 예약 : 예약 시작 시간 < 예약 종료 시간 - 연일 예약 : 오픈 시간 < 예약 종료 시간, 마감 시간 > 예약 시작 시간
	기본/최대 인원수 제한	최소 인원수 < 선택 인원수 < 최대 인원수
예약 가능 일정 확인	영업일/영업시간 확인	- RepeatDate에서 운영 요일과 예약 요일 비교 - 해당 요일의 RepeatDate의 운영 시간 범위와 예약 시간대 비교
	예약불가 일정 및 기존 예약 일정 확인	- 예약 시작 시간 < ExceptionDate의 시작 시간 < 예약 종료 시간 - 예약 시작 시간 < ExceptionDate의 종료 시간 < 예약 종료 시간
요금 계산	실 이용 시간 계산	- 하루 예약 : 종료 시간 - 시작 시간 - 연일 예약 : 일자별 총 운영 시간 + (시작일의 마감시간 - 시작일의 시작 시간) + (마감일의 종료 시간 - 마감일의 오픈시간)
	이용 시간 및 인원수 반영한 요금 계산	총 요금 = 예약 시간 * 시간당 요금 + 추가 요금(추가 인원 수)

## 실제 구현 화면 및 코드

### 예약 알고리즘 (S002)

#### 날짜 입력

예약 시작일

예약 종료일

시간 당 비용

추가 비용

총금액

예약

2020  
Tue,  
Sept 22

< SEPTEMBER 2020 >

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

CANCEL

Studio 이폰 수집

## 실제 구현 화면 및 코드

### 예약 알고리즘 (S002) - 코드

#### 날짜 입력 제한

```
// 2-1) 날짜 조건
if (this.start_date != "") {
    this.startTimes = this.setTime(this.startDay);
    if (this.startDate < todayTime) {
        this.start_date =
            this.today.getFullYear() +
            "-" + (this.today.getMonth() + 1) +
            "-" + this.today.getDate();
        alert("대여 시작일을 오늘 이후로 선택하세요.");
    }
}
if (this.end_date != "") {
    this.endTimes = this.setTime(this.endDay);
    if (this.startDate > this.endDate) {
        this.end_date = "";
        alert("대여 종료일을 시작일 이후로 설정하세요.");
    }
}
```

#### 시간 입력제한

```
// 2-2) 시간 조건
if (this.start_date != "") {
    if (this.end_date != "" &&
        this.start_date == this.end_date &&
        this.startTimes[this.startTimes.length - 1] == startTime) {
        this.start_time = 25;
        alert("대여 시작시간을 종료 시간 전으로 설정하세요. ");
    }
    //하루 예약이면 시작시간 < 종료시간
    if (startTime < 24 && startTime >= endTime) {
        this.end_time = 25;
        alert("대여 종료시간은 시작시간 이후로 설정하세요.");
    }
}
if (this.end_date != "") {
    if ((endTime < 24) & (this.endTimes[0] == endTime)) {
        this.end_time = 25;
        alert("대여 종료시간을 오픈 시간 이후로 설정하세요.");
    }
}
```



## 실제 구현 화면 및 코드

### 예약 알고리즘 (S002) - 코드

#### 비정규 휴일 확인

```
checkException() {
  if (this.startDayTime != "" & this.endDayTime != "") {
    for (var i = 0; i < this.exceptionLength; i++) {
      var exc_startOnlyDate = (this.schedule.exceptionDate[i].startDate).split(' ')[0];
      var exc_startOnlyTime = (this.schedule.exceptionDate[i].startDate).split(' ')[1];
      var exc_endOnlyDate = (this.schedule.exceptionDate[i].endDate).split(' ')[0];
      var exc_endOnlyTime = (this.schedule.exceptionDate[i].endDate).split(' ')[1];
      var exc_startDate = this.transTime(exc_startOnlyDate, exc_startOnlyTime);
      var exc_endDate = this.transTime(exc_endOnlyDate, exc_endOnlyTime);
      if ((this.startDayTime <= exc_startDate & exc_startDate < this.endDayTime) |
          (this.startDayTime < exc_endDate & exc_endDate <= this.endDayTime)) {
        return 0;
      } else {
        continue;
      }
    }
    return 1;
  }
},
```

#### 시간대 반영한 요금계산

```
var total_price = 0;
var difDate = (this.endDate - this.startDate) / (1000 * 60 * 60 * 24); //일자 차이
// 4-1) 예약 일자 사이에 날짜별 영업 시간 구하기 (for문)
var cntTime = 0;
if (this.start_date != "" & this.end_date != "" & this.start_time < 24 & this.end_time < 24) {
  if (difDate == 0) { //1일 예약
    cntTime = (endTime - startTime)
  } else { //연일 예약
    for (let i = 0; i <= difDate; i++) {
      var next = new Date(this.startDate + (i * 1000 * 60 * 60 * 24));
      var nextDay = this.transWeekDay(
        next.getFullYear() +
        "-" + (next.getMonth() - 1) +
        "-" + next.getDate());
      let j = this.repeatedDays.indexOf(week[nextDay], 0)
      this.openTime = parseInt(this.repeated[j].time.split('-')[0]);
      this.closeTime = parseInt(this.repeated[j].time.split('-')[1]);
      if (i == 0) { //시작일 : 마감시간-시작시간
        cntTime += (this.closeTime - parseInt(startTime));
      } else if (i == (difDate)) { //종료일: 종료시간-오픈시간
        cntTime += (parseInt(endTime) - this.openTime);
      } else { // 그 사이날짜 : if 영업일 >> 마감시간-오픈시간
        if (this.repeatedDays.indexOf(week[nextDay], 0) > -1) {
          cntTime += (this.closeTime - this.openTime)
        }
      }
    }
  }
}
```

## 실제 구현 화면 및 코드

### 검색 알고리즘

필터값/ 검색어 입력	정보 입력	- 카테고리, 날짜, 주소, 인원수, 최소가격 정보 입력
	‘인원수’ 예외 처리	- 인원수의 음수화 제한
F001 검색 종류 분류/ 검색	일반 검색	- 입력된 검색어를 포함한 모든 스튜디오 출력 - 검색 항목 : 회사명, 스튜디오명, 상세 설명, 주소, 태그
	태그 검색	- 검색어 앞에 “#”이 붙으면 태그 검색 메소드 작동 - 태그명과 검색어가 일치할 때만 출력
화면 처리	무한스크롤링	- 스크롤바가 바닥에 위치하면, 5개씩 화면에 출력
F003 정렬	정렬 방법	- 주어진 필터/검색어 조건 하에서 기준으로 다시 정렬 후 출력
	정렬 기준	- 인기, 가격 오름차순, 가격 내림차순, 평점순

## 실제 구현 화면 및 코드

### 검색 구현 화면

#### 필터/검색 화면 F001

내가 원하는 조건으로 찾기

카테고리로 찾기

예약 날짜로 찾기

전체

연도-월-일

날짜 초기화

주소로 찾기

도시명을 입력해주세요

시

구/면/읍 입력해주세요

구/면/읍

공간 크기로 찾기

최소면적을 입력해주세요

m2~

최대면적을 입력해주세요

m2

인원 규모로 찾기

-1

0명

+1

시간당 가격으로 찾기

\$

최소금액을 입력해주세요

원~

최대금액을 입력해주세요

원

초기화

적용

#### 정렬/결과 화면 F003

높은 가격순

**한국지도자아카데미**  
사무실 / 서울  
70~80년대 옛 서재  
681,000 원/시간  
평가 없음

**하슬라아트월드\_세미나실**  
편 / 강원  
통유리 너머로 들어오는 푸른 바다와 하늘이 매력적인 ...  
619,000 원/시간  
평가 없음

## 실제 구현 화면 및 코드

### 검색 코드

#### 필터/검색 코드 F001

```
<!-- 필터 검색 쿼리 | 이미지검색 쿼리 -->
<select id="selectStudioByFilter" parameterType="searchCon" resultMap="studioRM">
  <include refid="Select-J-Studio-Category-filter-tag-review" />
  <where>
    <choose>
      <!-- tag 검색 :: tag 검색 시 필터 조건 정리하지 않음 -->
      <when test="searchTag != null">
        AND t.tag_name = #{searchTag}
      </when>
      <otherwise>
        <!-- 이미지 추론 stuId 결과 조건 -->
        <if test="stuId != null">
          AND
          <foreach collection="stuId" item="id" open="(" close=")" separator="OR">
            s.stu_id = #{id}
          </foreach>
        </if>
        <if test="address1 != null or address2 != null">
          AND f.address Like "%${address1}%"
          AND f.address Like "%${address2}%"
        </if>
        <if test="minSize != null">
          AND f.size >= #{minSize}
        </if>
        <if test="maxSize != null">
          AND #{maxSize} >= f.size
        </if>
        <if test="minUnitPrice != null">
          AND f.unit_price >= #{minUnitPrice}
        </if>
        <if test="maxUnitPrice != null">
          AND #{maxUnitPrice} >= f.unit_price
        </if>
        <if test="capacity != null">
          AND f.max_capacity >= #{capacity}
        </if>
        <if test="searchContent != null">
          <!-- 검색어 : 회사이름, 스튜디오 이름, 상세설명, 태그, 주소 -->
          AND
          <foreach collection="searchContent" item="search" open="(" close=")" separator="OR">
            s.name Like '%${search}%'
            OR c.category_name Like '%${search}%'
            OR s.description Like '%${search}%'
            OR t.tag_name Like '%${search}%'
            OR f.address Like '%${search}%'
            OR com.name Like '%${search}%'
          </foreach>
        </if>
      </choose>
    </where>
  </select>
```

#### 정렬 코드 F003

```
<!-- 정렬 1:인기순(기본), 2: 가격내림차순, 3: 가격오름차순 4: 평점순 -->
<if test="orderCon == 1">
  ORDER BY res.count DESC
</if>
<if test="orderCon == 2">
  ORDER BY f.unit_price DESC
</if>
<if test="orderCon == 3">
  ORDER BY f.unit_price ASC
</if>
<if test="orderCon == 4">
  ORDER BY rv.avg DESC
</if>
limit #{page} , 5
```

## 실제 구현 화면 및 코드

이미지 검색 - 배경 F002


- 촬영공간 Domain 특성 상, 텍스트만으로 사용자가 원하는 검색 결과 도출 제한적
  - 촬영공간 Domain은 공간의 밝기, 배치 등 텍스트로 표시될 수 없는 조건의 영향력이 큼
- 촬영공간 Domain 특성을 고려하여 이미지를 통한 유사 스튜디오 검색 메소드 사용



## 실제 구현 화면 및 코드

### 이미지 검색 알고리즘 F002

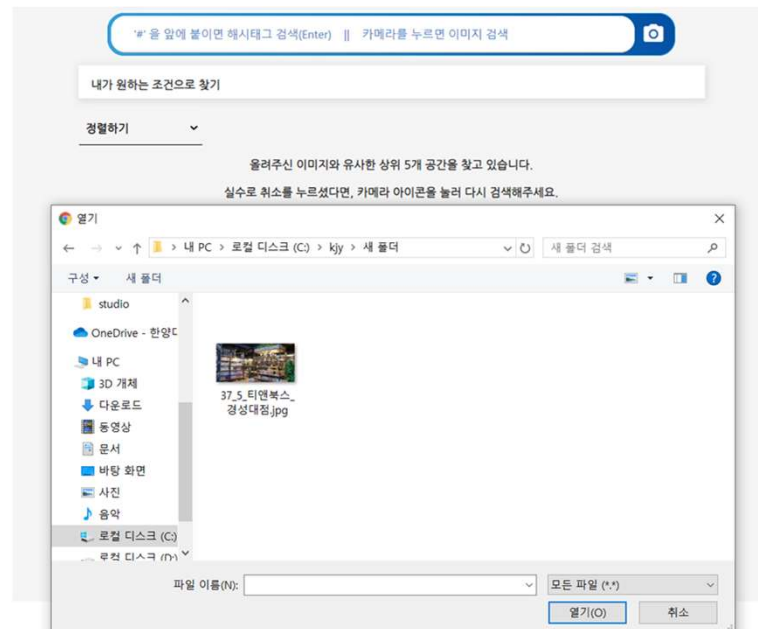
이미지 입력	- 사용자 로컬의 이미지를 서버에 등록
이미지 특징 추출	- ResNet152 모델을 활용해 특징 추출
특징 간 유사도가 높은 5개 이미지명 추출	- 서버의 스튜디오 Main Image 특징들과 유사도 도출 후 상위 5개 이미지명 추출 - Cossine 유사도 활용 - 빠른 검색 속도를 위해, 서버 상 이미지들은 미리 특징 추출, 정기적으로 업데이트.
상위 5개 스튜디오 ID 추출	- 이미지명을 바탕으로, 유사도가 가장 높은 이미지를 소유한 상위 5개 스튜디오의 ID 추출
추출된 ID에 해당하는 스튜디오 출력	- 다시 DB에서 해당 ID를 가진 스튜디오와 필요 정보를 리스트에 출력



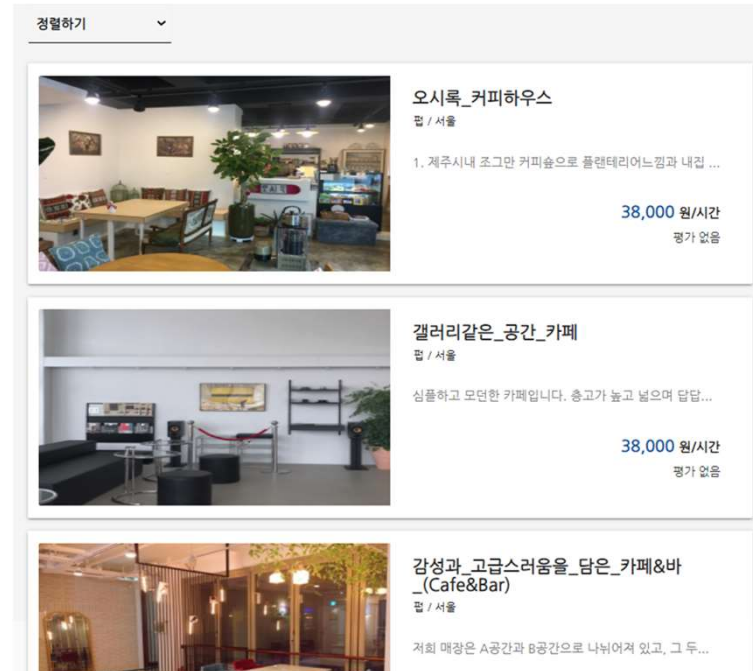
## 실제 구현 화면 및 코드

### 이미지 검색 구현 화면 F002

#### 이미지 입력 화면



#### 결과 화면



## 실제 구현 화면 및 코드

### 이미지 검색 코드 F003

#### 특징 추출 코드

```
# CNN 모델 불러오기 __ resnet152
def get_model():
    model = torch.hub.load('pytorch/vision:v0.6.0', 'resnet152', pretrained=True)
    return model

# In[3]:

# feature 추출하기
def extract_feature(saved_img_path, image_name):
    input_image = Image.open(join(saved_img_path, image_name))
    model = get_model()
    preprocess = transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])
    input_tensor = preprocess(input_image)
    input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected by the mode

    if torch.cuda.is_available():
        input_batch = input_batch.to('cuda')
        model.to('cuda')

    with torch.no_grad():
        output = model(input_batch)

    return torch.nn.functional.softmax(output[0], dim=0)
```

#### 유사도 높은 이미지명 추출 코드

```
def get_sim_image_names(saved_img_path, file_name, simpath):
    target_img = extract_feature(saved_img_path, file_name)
    feature_list = get_sim_image(simpath)
    sim_list = []
    cos = torch.nn.CosineSimilarity(dim=-1)
    for feature in feature_list:
        sim_degree = cos(target_img, feature['vector'])
        sim = {
            'name' : feature['name'],
            'sim_degree' : float(sim_degree)
        }
        sim_list.append(sim)
    df = pd.DataFrame(sim_list)
    image_names = df.sort_values(by='sim_degree', ascending=False)[:50]['name']
    return image_names
```



## 실제 구현 화면 및 코드

### 추천 알고리즘

1.  
최근 한 달 간  
인기 상품 추천

2.  
무관심 아이템을  
활용한  
Collaborative  
Filtering  
추천

3.  
자연어 기반  
유사 스튜디오  
추천

## 실제 구현 화면 및 코드

추천 알고리즘 - 기술 사용 배경 1 - CF 문제와 솔루션

### Cold Start 문제

다음과 같은 항목에서 추천 불가

- 1) 한 번도 평가하지 않은 유저
- 2) 한 번도 평가받지 않은 아이템



인기 스튜디오 추천

### Data Sparsity 문제

User-Item Matrix가  
대부분 비어있음  
추천 정확도 감소




무관심 상품을 활용한 CF 추천

## 실제 구현 화면 및 코드

### 추천 알고리즘 - 1) 인기상품 추천 알고리즘 M002

로그인 여부 확인	- Session Storage에서 로그인 여부를 확인한다
리뷰 여부 확인	- 리뷰 평점을 한 번이라도 평가했는지 확인한다 - 로그인과 리뷰 둘 중 하나라도 했으면 CF 알고리즘으로 넘어간다
인기순 정렬	- 최근 한 달 간 예약이 가장 많은 순으로 정렬하여 리스트를 출력한다
상위 8개 출력	- 상위 8개 스튜디오를 메인 페이지 추천란에 출력한다



## 실제 구현 화면 및 코드

### 추천 알고리즘 - 1) 인기상품 추천 코드 M002

#### 인기 상품 추천 코드

```
# 메인 추천 업체 가져오기
def get_ranked_studio() :
    sql = "SELECT s.stu_id, s.name, c.category_name, f.address, f.unit_price, s.main_img "
    sql += "FROM studio s "
    sql += "JOIN studio_filter f "
    sql += "ON s.stu_id = f.stu_id "
    sql += "JOIN studio_category c "
    sql += "ON s.category_id = c.category_id "
    sql += "LEFT OUTER JOIN "
    sql += "(SELECT stu_id, COUNT(stu_id) count FROM reservation WHERE res_date > SUBDATE(NOW(), INTERVAL 1 MONTH) GROUP BY stu_id) res "
    sql += "ON s.stu_id = res.stu_id "
    sql += "order by res.count desc "
    sql += "limit 8"

    result = db.run_query(sql)
    return result
```

## 실제 구현 화면 및 코드

### 추천 알고리즘 - 2) 무관심 상품을 활용한 CF - 이론적 배경 M003

#### 1. 평가되지 않은 대부분이 무관심 상품

이영남, 김상욱. 2017. 추천시스템에서의 데이터 임пут레이션 분석. *정보과학회논문지 제44권 제 12호* pp.1333-13371

#### 2. 무관심 상품을 골라 0점을 부여했을 때, 추천 정확도 상승

황원석. 2016. 추천 시스템 개선을 위한 무관심 아이템, 신뢰 네트워크, 카테고리 전문가 활용방안

#### 3. 무관심 항목에 대해 2점을 부여했을 때 높은 정확도가 나타남

이영남, 김상욱. 2017. 추천시스템에서의 데이터 임пут레이션 분석. *정보과학회논문지 제44권 제 12호* pp.1333-1337

#### 4. 무관심 상품 선택 논리는 아래의 논문을 인용하여 알고리즘

Hyung-Ook Kim. 2017. An Efficient and Effective Method to Find Uninteresting Items for Accurate Collaborative Filtering

## 실제 구현 화면 및 코드

### 추천 알고리즘 - 2) 무관심 상품을 활용한 CF M003

#### 무관심 상품 찾기

##### 기본 논리

- 확률을 기반으로 하여 무관심 상품 선택
- 회원 중 가장 해당 도메인에 관심이 없는 (리뷰를 하지 않은) 유저가 평가하지 않은 상품 중 가장 인기 있는 (리뷰 수가 가장 많은) 상품을 확률적으로 선택

##### 유저 선택 확률

$$P(u_i) = \frac{|I| - |I(i)|}{\sum_{u_k \in U} (|I| - |I(k)|)}$$

- 모든 상품군,  $I(i)$ : 유저  $i$ 가 평가한 상품군,  $U$ : 모든 유저
- 평가가 적을수록 선택될 확률이 높음

##### 아이템 선택 확률

$$P(i_j) = \frac{|U(j)|}{\sum_{i_k \in I(s)} |U(k)|}$$

- $U(j)$ : 상품  $j$ 를 평가한 유저군
- 상품  $j$ 는 평가를 많이 받을수록 선택될 확률이 높음

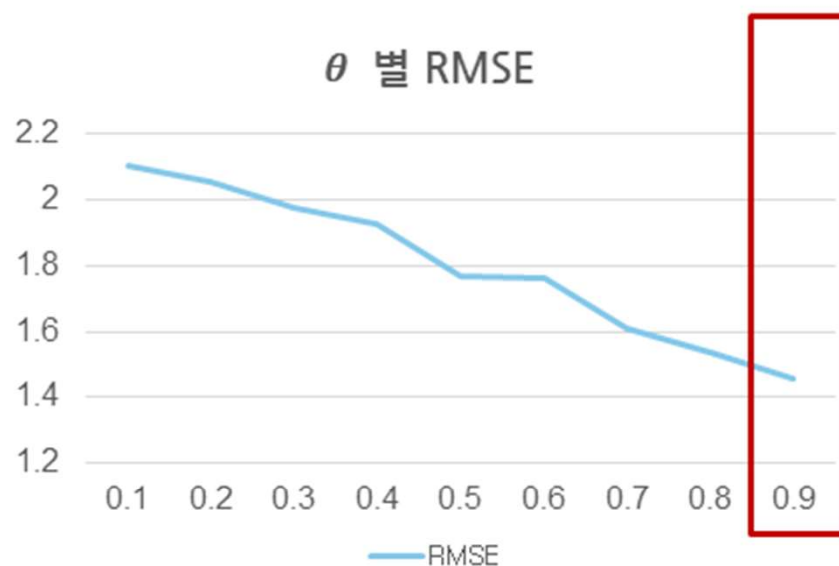
## 실제 구현 화면 및 코드

### 추천 알고리즘 - 2) 무관심 상품을 활용한 CF M003

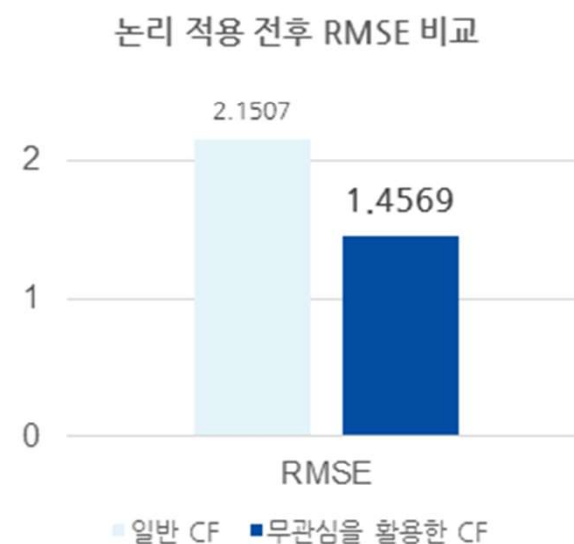
무관심 상품 찾기	선택 조건	<ul style="list-style-type: none"> <li>- <math>\sum_{k=1}^{i-1} P \leq n_{rand} &lt; \sum_{k=1}^i P</math></li> <li>- <math>n_{rand}</math> : 0~1 사이의 난수</li> <li>- 확률 P가 클수록 선택될 확률이 높음. 유저와 아이템에 반복</li> <li>- 같은 유저, 같은 항목이 반복되어 선택되는 것을 방지하기 위함</li> </ul>
무관심 상품에 2점 할당		<ul style="list-style-type: none"> <li>- 1~5 척도의 리뷰 점수를 2점으로 할당</li> </ul>
반복	반복 조건	<ul style="list-style-type: none"> <li>- 결측치 <math>\theta\%</math> 만큼 할당할 때까지 반복</li> <li>- <math>\theta</math>는 데이터셋마다 다르며 연구자가 결정 (Hyper Parameter)</li> </ul>
Item-Based CF로 빈 부분 예측		<ul style="list-style-type: none"> <li>- 채우고 남은 부분을 Cossine 유사도를 통해 예측</li> </ul>
상위 8개 스튜디오 추천		<ul style="list-style-type: none"> <li>- 해당 유저가 평가하지 않은 스튜디오 중 예상 점수가 높은 스튜디오 8개 추천</li> </ul>

## 실제 구현 화면 및 코드

추천 알고리즘 - 2) 무관심 상품을 활용한 CF - 성능 향상 M003



→  $\theta$ 를 0.9로 지정



→ 32.23% 감소



## 실제 구현 화면 및 코드

추천 알고리즘 1), 2) 화면 M002 M003

### 추천 결과 화면

추천 인기공간

[더보기](#)



블롱드셰프 작은식당(작당)  
스튜디오/ 대구

120,000 원/시간



후오비 블록체인  
스튜디오/ 전주

50,000 원/시간



질은 원목  
스튜디오/ 경기

90,000 원/시간



라오테아로아 스튜디오  
스튜디오/ 경기

130,000 원/시간



## 실제 구현 화면 및 코드

### 추천 알고리즘 - 2) 무관심 상품을 활용한 CF - 코드 M003

#### User 선택

```
# user_selection
def user_selection(ratings):
    # (1) probability of selection
    tot_nan_u = ratings.isnull().sum().sum()
    prop_u = pd.DataFrame(ratings.isnull().sum(axis=1)/tot_nan_u, columns=['Pui'])
    # print("user total nan : {}".format(tot_nan_u))

    # (2) select_user
    nrand = np.random.rand()
    prop_u['condition'] = nrand+prop_u['Pui']-1
    selected_u=prop_u[(prop_u['Pui']>prop_u['condition']) & (prop_u['condition']>=0)].index
    # print(prop_u)
    # print("seleced_u : {}".format(selected_u))
    return selected_u
```

#### 예측 점수 출력

```
# 전체 예측점수 테이블 출력
def pred_scores(ratings,table):
    ratings_combinations = list(product(ratings.index, ratings.columns))
    rating_list = []
    for item, user in ratings_combinations:
        score = pred_score(item,user,ratings,table)
        rating_predict = {
            'user': user,
            'item': item,
            'score': score
        }
        rating_list.append(rating_predict)
    rating_list = pd.DataFrame(rating_list)
    rating_table = pd.pivot_table(rating_list,index='user',columns='item',values='score')
    return rating_table
```

#### Item 선택 & 2점 할당

```
# item_selection
def item_selection_injecton(ratings, selected_u,init_nan,theta, theta_d):
    flag = True
    result=[theta, flag]
    # (1) probability of selection
    for i in selected_u:
        tot_nan_i = ratings.isnull().sum().sum()
        candi_item = ratings.loc[i,:].index
        down_df= ratings.loc[:,candi_item[candi_item.isnull()]].index
        down_df_T = np.transpose(down_df)

        # print(down_df_T)
        tot_nan_i = down_df_T.notnull().sum(axis=1).sum()
        # print("tot_nan : {}".format(tot_nan_i))
        prop_i = pd.DataFrame(down_df_T.notnull().sum(axis=1)/tot_nan_i, columns=['Pij'])
        nrand = np.random.rand()
        prop_i['condition'] = nrand+prop_i['Pij']-1
        selected_i=prop_i[(prop_i['Pij']>prop_i['condition']) & (prop_i['condition']>=0)].index

        # print(prop_i)
        if len(selected_i)>0 :
            print('len : ',len(selected_i))
            ratings.loc[i,selected_i] = 2
            tot_nan = ratings.isnull().sum().sum()
            theta = tot_nan / init_nan
            result[0] = theta
            print("init : {}, tot : {}".format(init_nan, tot_nan))
            if theta <= (1-theta_d) :
                result[1]=False
                break
            print("theta : {}".format(theta))
    return result
```

## 실제 구현 화면 및 코드

자연어 기반 유사 스튜디오 추천 (S003) - 기술 사용 배경

### 유저 관심 정보

스튜디오 페이지 방문으로  
유저 관심 항목이 간접적으로 표현

### 스튜디오 정보의 정형성

스튜디오 설명과 tag 정보 정형성과  
객관성이 높아 활용 용이



스튜디오 정보만으로 현재 탐색 중인 스튜디오와  
유사한 스튜디오 추천 알고리즘 구현의 조건 확보

## 실제 구현 화면 및 코드

### 자연어 기반 유사 스튜디오 추천 (S003)

데이터 정제 (NLP)	1) Tokenization	<ul style="list-style-type: none"> <li>- description : 정규화, corps 단위로 Noun, Adjective 위주로 <b>tokenization</b></li> <li>- tag : 복합명사도 형태소 단위로 <b>tokenization</b>.</li> </ul>
	2) 불용어 및 제외 단어	<ul style="list-style-type: none"> <li>- 글자수 2자 미만의 단어 제외</li> <li>- 예약 및 문의 관련 표현과 불필요한 Adjective의 어미는 <b>불용어</b>로 제외</li> </ul>
	3) 새로운 태그 컬럼 생성	<ul style="list-style-type: none"> <li>- Tag1 : 기존 태그와 description의 형태소 합집합. <b>model의 dataset</b></li> <li>- Tag2 : description에도 포함되는 태그(<b>CoreTag</b>)</li> <li>+ 태그가 없다면 description의 형태소로 대체</li> <li>+ 2경우 모두 아니면 기존 태그 사용</li> </ul>
유사 단어 (Word2Vec)	4) word2vec skip-gram모델	<ul style="list-style-type: none"> <li>- 다수의 단어 확보한 Tag1을 dataset으로 사용</li> <li>- 정확도가 높은 Tag2를 기준으로 유사 단어 추출</li> <li>- 모델에 반영할 단어의 <b>최소 빈도 기준</b>을 낮춰 빈도가 낮아도 의미 있는 단어 누락 방지</li> <li>- word2vec의 <b>Skip-gram</b> 모델 사용하여 단어 임베딩</li> </ul>
	5) 유사 키워드 컬럼 생성	<ul style="list-style-type: none"> <li>- Tag2의 태그 별로 유사도 상위 4개 단어 선정하여 <b>Extend_Tag</b>컬럼 설정</li> <li>- Tag2의 태그 개수에 따라 추출할 유사 단어수 차별화하여 태그 개수에 대한 의존도 절감</li> </ul>
빈도수 계산	6) 키워드 빈도로 유사 스튜디오 선정	<ul style="list-style-type: none"> <li>- 타겟 스튜디오의 Tag2 와 유사 키워드를 나머지 스튜디오가 얼마나 많이 포함하고 있는지를 기준으로 스튜디오 간의 유사성 판별.</li> <li>- tag2는 가중치 1로, 유사키워드는 유사도 평균으로 가중치 조정하여 빈도수 산출</li> </ul>

## 실제 구현 화면 및 코드

자연어 기반 유사 스튜디오 추천 (S003)

유사 스튜디오 리스트 화면

### 유사한 스튜디오 보기



R401

펍/ 경기

38000 원/시간



영등포구\_모던하고...

집/ 서울

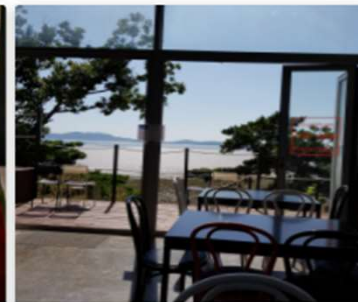
56000 원/시간



북합문화공간\_코앰커피

카페/ 서울

38000 원/시간



여름\_특가\_촬영장...

펍/ 인천

50000 원/시간



## 실제 구현 화면 및 코드

### 자연어 기반 유사 스튜디오 추천 (S003) - 데이터 set

단어 빈도	테스트 단어	1번모델					2번모델				
		1순위	2순위	3순위	4순위	5순위	1순위	2순위	3순위	4순위	5순위
108개	가정	주차	거실	사진	빈티	가구	카페	촬영	사용	가능	느낌
194개	사진	빈티	주차	개인	원룸	아파트	촬영	아파트	느낌	카페	사용
873개	촬영	분위기	자연	거리	감성	컨셉	느낌	위치	공간	사용	이용
76개	원목	근처	연출	카페	문의	주차장	촬영	공간	이용	화이트	분위기
311개	아파트	사진	거실	가구	주차	미니	촬영	공간	이용	위치	사용
61개	루프	주택	추가	운영	바로	모임	촬영	공간	사용	카페	위치
220개	자연	촬영	제품	감성	거리	컨셉	촬영	사용	공간	느낌	소품
133개	채광	주방	주변	테라스	경우	원룸	촬영	이용	가능	공간	위치
298개	스튜디오	위치	소품	공간	대여	영상	공간	사용	위치	촬영	아파트
94개	광고	주차	채광	테라스	경우	사진	촬영	서울	공간	이용	느낌
331개	거실	주차	가구	아파트	사진	미니	분위기	촬영	느낌	자연	사용
280개	주택	야외	건물	루프	침실	운영	촬영	공간	위치	가능	분위기
99개	파티	카페	서울	위치	스튜디오	감성	느낌	아파트	촬영	공간	사용
66개	빌라	주택	인테리어	운영	바로	가능	촬영	분위기	공간	아파트	위치
141개	카페	마당	서울	위치	근처	햇살	촬영	사용	느낌	위치	공간
69개	빈티	사진	주차	원룸	마당	하우스	촬영	사용	장소	느낌	카페
9개	플라워	가능	루프	샤워	엘리베이터	책상	모던	샤워	장소	루프	주차장
66개	홍대	카페	자연	촬영	서울	감성	촬영	사용	화이트	자연	이용
678개	공간	이용	주차장	연출	예약	위치	촬영	위치	사용	소품	주택
202개	화이트	분위기	컨셉	촬영	하우스	서울	촬영	이용	공간	사용	서울
263개	느낌	사용	조명	내부	건물	전체	촬영	사용	공간	카페	위치
224개	분위기	촬영	배경	거리	컨셉	실내	촬영	이용	공간	주택	위치
50개	원룸	사진	주택	공원	채광	주방	주택	촬영	가능	분위기	느낌
62개	사무실	문의	공원	카페	서울	위치	촬영	느낌	위치	전체	사용
153개	서울	햇살	카페	마당	위치	하우스	촬영	공간	느낌	사용	화이트
85개	마당	카페	서울	근처	정원	위치	카페	촬영	느낌	사용	빈티
92개	정원	소품	시간	위치	마당	카페	촬영	소품	위치	아파트	이용

\*\* 테스트 단어 별로 유사 단어가 중복되는 경우 파란색 배경으로 처리

#### Tag1

- description와 태그의 단순 합집합
- Tag1로 학습할 경우 2번보다 다양한 단어 제시, 다만 연관성 낮은 단어도 포함

#### Tag 2

- 비교적 태그 중심으로 추출
- Tag2로 학습할 경우 연관성이 높은 단어 위주로 추출하지만, 유사 단어 set이 제한적



Tag1을 모델 학습하여 다양한 유사 단어 확보  
Tag2를 기반으로 유사 단어 추출하여 정확성 확보



## 실제 구현 화면 및 코드

### 자연어 기반 유사 스튜디오 추천 (S003) - 데이터 set 코드

#### [자연어 처리기]

```
def getDescList(dataset):
    n=len(dataset)
    okt=Okt()
    descList=[]
    stop_words=["문의", "가능", "가능합니다", "요일", "운영", "마감", "요일", "시간",
                "습니다", "주세요", "입니다", "있는", "메일", "또는",
                "아주", "필요합니다", "저희", "일요일", "주말", "토요일", "가능하며",
                "있습니다", "적합합니다", "대여", "종갓습니다", "무인", "부터",
                "최대", "최소한", "원하는", "위치", "최대한", "원하시면",
                "원하시는", "가능하세요", "아니라", "원하신다면",
                "위해", "화장실", "직원"]
    stop_words=list(set(stop_words))

    for i in range(n):
        tempList=[]
        descList.append(dataset.loc[i, "description"]) # 한글자 이상만 뽑음
        descList.append(descList.replace("#n", ""))
        descList.append(descList.replace("#r", ""))
        descList.append(descList.replace("[^ㄱ-ㅎㅌ-ㅣ가-힣A-Z09]", ""))
        tempList=[j[0] for j in okt.pos(descList)
                  if ((len(j[0])>1)
                      &((j[1]=="Adjective")|(j[1]=="Noun"))
                      &(j[0] not in stop_words))]
        tempList=list(set(tempList))
        descList.append(tempList)
    return descList
```

불용어 사전

Tokenization

댓글/리뷰 분석에 강력한 Okt 자연어 처리기 사용

#### [데이터 셋 설정]

```
def getCoreTags2(dataset):
    n=len(dataset)
    coreTagList=[]
    okt=Okt()
    descList=getDescList(dataset) # 한글자 이상만 뽑음
    tags=getTagList(dataset) # tag를 한번 더 파싱
    for k in range(len(descList)):
        coreTags=[]
        # 태그가 description 단어에도 있는지 확인
        for t in range(len(tags[k])):
            if (len(tags[k][t])>0 and (tags[k][t] in descList[k])):
                coreTags.append(tags[k][t])
        # 1) 태그 수가 없으면 디스크립션으로
        if (len(tags[k])<=3):
            coreTagList.append(list(set(descList[k])))
        # 2) desc와 일치하는 태그가 적고 기존 태그 많으면 기존 태그 사용
        elif ((len(coreTags)<=3) and (len(tags[k])>3)):
            coreTagList.append(tags[k])
        # 3) 디스크립션과 일치하는 태그가 많으면 일치태그 사용
        elif (len(coreTags)>3):
            coreTagList.append(coreTags)
        else:
            coreTagList.append(list(set(descList[k])))
    return coreTagList
```

Tag 보정

## 실제 구현 화면 및 코드

### 자연어 기반 유사 스튜디오 추천 (S003) - 모델 코드

#### [모델 설정]

```
def wordVec(CoreTagData1):  
    size_num = 350      # Word vector dimensionality  
    min_word_count = 2  # Minimum word count  
    num_workers = 2     # Number of threads to run in parallel  
    context = 4         # Context window size  
    downsampling = 0.001 # Downsample setting for frequent words  
  
    model = gensim.models.Word2Vec(CoreTagData1,  
                                   sg=1,  
                                   workers=num_workers,  
                                   size=size_num,  
                                   min_count = min_word_count,  
                                   window = context,  
                                   sample = downsampling)  
  
    return model
```

#### sg (skip-gram 모델)

- 특정 1개 단어로 주변에 나올 단어를 예상하는 word2Vec 임베딩 모델.
- 단어 임베딩과정에서 단순히 동반 출현 빈도 뿐만 아니라 의미론적으로 관련 높은 단어 추출
- 다른 모델인 cbow보다 일반적으로 skip-gram의 성능이 우수.

#### window

- 특정 단어를 유추할 주변 단어의 수.
- Description 내용이 길지 않아 앞뒤로 2개 단어씩 고려하도록 window=4로 설정

#### min count

- 단어의 최소 빈도. 단어 누락을 방지하고자 2회 이상 등장한 단어 모두 포함